الشكل الأقدم للأحداث و مُتعهِّداتهن events and events handling في إبداع و أسباب تغييره

م. وائل حسن- أبو إياس

مقدمة

هذا المقال يُعتبَر تكملةً لمقالٍ سابقٍ كتبتُه عن الطرق المُختلفة لدعم "الأحداث events" و "مُتعهّداتها events handlers" في بعض لغات البرمجة (أعني بهن الـ: java و visual basic.net و C#). ثم قارنتُ حينها بين كل تلك الطرق و بين الطريقة التي يتم بها الدعم حالياً في إبداع؛ بغرض البرهنة علي قوة و بساطة قواعد الأخيرة التي قد لا تظهر عند التأمل البسيط.
لذلك فأنا أفضِّل قراءة ذلك المقال السابق في البداية. ثم يمكن قراءة هذا المقال بعده مباشرةً لتسهيل الربط بين ما فيهما من معلوماتٍ و استنتاجاتٍ و تلميحات. و كذلك لضمان وصول نظرتِي الشاملة فيما يخص الأحداث و تَعَهُّدها إلي القاريء الكريم، و أن يستوعب كيفية خضوع تلك النظرة للقواعد الأصولية التي وضعتُها في كتابي "رسالة البرمجة بإبداع" عن منهجي فيما يخص تصميم لغات البرمجة.

***

قلتُ من قبل في أثناء حديثي عن "الوراثة الجزئية partial inheritance" أنني حينما كنتُ أقوم بتصميم لغة البرمجة "إبداع" في الفترة الأولي من حياة مشروع البرمجة بإبداع (أي منذ حوالي العامين و النصف): كان تصميمها المبدئي يختلف للغاية عن التصميم النهائي الذي استقرّت عليه حالياً، و قلتُ كذلك أن إبداع شهدت علي الأقل شكلين مختلفين تمام الاختلاف عن بعضهما البعض، لدرجة أنه يمكن اعتبار كل واحدٍ منهما لغةً مختلفةً قائمةً بذاتها !؛ لأنني كنتُ في كل فترةٍ أعيد تقييم آرائي العلمية بما يتناسب مع ما حصّلتُه من معرفةٍ جديدة، و من ثم أقوم بإعادة تقييم التصميم السابق بما يتفق مع ما يستجد عندي من آراءٍ صرتُ أقتنع بها بعد أن كنتُ أري ما يُخالفها.

و أثناء تلك الفترات التي كانت تتغير فيها قَناعاتي وجدتُ أن هناك بعض المُكوِّنات (أعني: قواعد و تعبيرات) كنتُ مقتنعاً بأهميتها في الأشكال القديمة من إبداع، و لكن بعد التفكير الجيد فيها وجدتُ أنه من الأفضل ألا يتم ضمها إلي اللغة في نسختها النهائية. و كان من بين تلك المُكوِّنات التي تم التخلي عنها ما أطلقتُ عليه اسم "الوراثة الجزئية partial inheritance" أو "الوراثة الناقصة incomplete inheritance" التي تحدثتُ عنها في مقالٍ وافٍ من قبل.

و كذلك هناك مُكوِّنَي "الحَدَث event" و "مُتَعَهِّد الحدث event handler" كمُكوِنَيْن قائمَيْن بذاتيهما، و هما اللذان سأخصِّص لهما هذا المقال لبسط الحديث عنهما. و فيما يلي شرحٌ للقواعد القديمة لكليهما في لغة إبداع (قبل الإعلان عن المشروع رسمياً)، و هي القواعد التي تم التخلص منها بعد ذلك في التنقيحات المتتالية لتصميم اللغة، ثم سأرفق شرحاً بالأسباب التي جعلتني أُبعِدهن في النهاية.

و سُتلاحظون أن في تلك القواعد بعض الأمور الأخري التي تم إزالتها بدورها من مواصفات اللغة، مثل خصائص "اللزومية" و "الجِدّة"، و شرحهما بتلخيصٍ سيكون موجوداً في الصفحات القادمة بمشيئة الله تعالي و بالتالي سيكون من اليسير فهم معانيهن و تأثيراتهن. و حتي إن لم يكن بالإمكان فهمهن من مجرد ذلك الشرح المُختصَر فإن ذلك لن يؤثر علي القدرة علي فهم الأمور الرئيسة في هذا المقال: و التي هي "الأحداث و مُتعهّداتها بشكلهما القديم في لغة إبداع".

و سأقوم هنا بنقل القواعد التي كتبتُها بخصوص المُكوِّنين كما كنتُ قد كتبتُها من قبل في المُواصفات القياسية لإبداع قبل التغييرات النهائية، علي أنني قد قمتُ بإعطاء نفسي الحرية التامة في التعديل في تلك الشروحات القديمة؛ لفك بعض الغوامض أو تيسير بعض الصعوبات أو لحذف بعض الأشياء التي ستصعب استيعاب الفكرة الرئيسة هنا (حينما يكون حذفها لا يكون مشكلةً في حد ذاتها).

قبل البدء بإيراد القواعد أود التنبيه علي شيءٍ هام: أنه لفهم معظم ما سيتم شرحه هنا بشكلٍ واضحٍ فيجب أن يكون القارئ الكريم علي درايةٍ (و لو جزئيةٍ) بقواعد لغة إبداع. كما أن هناك بعض الأمور التي أظن أنه لا سبيل لفهمها إلا لمن استوعب الكثير من قواعد إبداع و الأصول المنهجية التي أعمل بناءً عليها.

الحدث event

نبذة بسيطة:

هو مُكوّنٌ يُستخدَم لتعريف مجموعةٍ من الشروط المنطقية، على أساس أنها تُمثّل وحدةً منطقيةً واحدة. فإذا ما تَحقّقَت هذه الشروط مُجتمعةً: قلنا أن الحدث المُؤلّف منها قد وَقَع.

و قد يكون مِن ضِمن هذه الشروط المنطقية اشتراط وقوع حدثٍ أو أحداثٍ أخرى، فإذا ما وقعت تلك الأحداث الأخري و تحققَتْ باقي الشروط: يكون الحدث المُكوّن مِن كل هذا قد تحقق.

و يمكننا كذلك إعلام البرنامج أن الحدث قد وقع حتى إن لم يتم تحقق الشروط المكونة له.. و شرح جميع ذلك فيما يلي.

الصياغة:

حدث اللزومية مستوى.الوصول الاستنساخ التغطية الجدة اسم.الحدث: \ مجموعة الشروط و الأحداث الأخرى المُوصِّفة للحدث / يرسل: \ القيم التي يُرسِلها هذا الحدث عند وقوعه إلى مُتعهّده (سيأتي شرح معني "المُتعهّد" فيما بعد) ليستخدمها في عملية التعهد. وتُكتَب كما في تعريفات الدخل و الخرج الخاصين بالإجراءات ما عدا أنه يمكننا كتابة قيم تُمرَّر كما هي إلى المتعهدات /

اللزومية:

الصفة	الشرح
لازم	هو الحدث الذي لا يقبل استخدام الوراثة الجزئية معه. أي لا يقبل أن يتم استثناؤه من عملية وراثة صنفه الحاوي له
بلا صفة (الافتراضية)	هو الحدث الذي يقبل الاستثناء عند الوراثة الجزئية لصنفه

مستويات الوصول:

المستوى	الشرح
عام (الافتراضي)	يمكن الوصول إليه من أي مكان
داخلي	يمكن الوصول إليه من داخل صنفه و الأصناف الوارثة له فقط
خاص	يمكن الوصول إليه من داخل صنفه الحاوي له فقط

الاستنساخ:

الصفة	الشرح
مجرد	ليس فيه كود بل إعلان فقط
مشترك	لا تصنع منه نسخة لكل كائنٍ object مُستنسَخ، بل يُستخدم مباشرةً باسمه
بدون صفة (الافتراضية)	هو الحدث المُستنسَخ العادي

التغطية:

الصفة	الشرح
سقف	لا يمكن تغطيته في الأصناف التي ترث صنفه
بدون صفة (الافتراضية)	هو الحدث العادي الذي يقبل التغطية

الجِدّة:

الصفة	الشرح
جديد	هو حدثٌ له نفس الاسم الخاص بحدث آخر في صنف يرثه صنفه و يقوم هو بالتعديل في بنائه implementation. أي أنه يُغطّي overrides حدثاً آخراً في صنفٍ أبٍ لصنفه (يمكن استخدام هذه الصفة حتى إذا تم استخدام الصفة مجرد من قبل نظراً لأنه في كل الأحوال لن يرى المبرمج إلا الحدث الجديد المُغطِّي للقديم)
بدون صفة (الافتراضية)	هو الحدث العادي الذي ليس هناك حدثٌ آخرٌ له ذات اسمه في الصنف أو الأصناف التي يرثها صنفه

مثال 1 :

```
حدث عام مشترك جديد تغير.الحساب:
    الحدث.الأول و الحدث.الثاني و            |
    س>10  أو ص <= 15 و                     |
    ع = ليس غ                              |
    أو                                     |
    س+ص= 25
    يرسل:
        رقم س×2  رقم ص+3
```

في هذا المثال سيعتبر البرنامج أن الحدث **تغير.الحساب** قد وقع في الحالات التالية:

1) وقوع الحدثين المُسميَيْن (الحدث.الأول) و (الحدث.الثاني) و في نفس الوقت تحقق الشرط س>10 ، أو:

2) تحقق الشرط ص>15 و في نفس الوقت تحقق الشرط ع = ليس غ ، أو:
3) تحقق الشرط س + ص = 25

و سيتم نداء مُتعهِّد الحدث بعدما يقع الحدث نفسه، و حينها سيتم تمرير القيمتين (س×2) و (ص+3) على أنهما قيمتين من النوع رقم بحيث يستخدمهما المتعهد إذا ما رغب في هذا.

مثال 2:

```
حدث عام مجرد تغير.الحساب
```

الاستخدام:

يتم إعلان وقوع الحدث إما:

1) عن طريق البرنامج الذي يتحقق بصورةٍ دوريةٍ من وقوع الأحداث المكتوبة داخله، و ذلك تلقائياً دون تدخل المبرمج، أو:
2) عن طريق المبرمج الذي يُعلِن وقوع الحدث بكتابة اسمه كأنه يستدعى إجراءً عادياً (فيما عدا عدم وجود قوسَيْ مُعامِلات الإجراء)، أي بالصياغة التالية:

```
اسم.الحدث
```

مثال:

```
لو س < ص:
    تغير.الحساب
    \ هكذا سيتم إعلام البرنامج بأن الحدث (تغير.الحساب) قد وقع بالفعل حتى و إن لم يتحقق مجموع شروطه التي كتبناها في تعريفه السابق /
```

مُتعهِّد الحدث event handler

نبذة بسيطة:

هو تعبيرٌ يُستخدَم لربط إجراءاتٍ مُعيّنة بحدثٍ مُعيَّن، بحيث يتم استدعاء تلك الإجراءات فور وقوع الحدث.

الصياغة:

في إبداع يقوم المبرمج بالربط بين الحدث (أو الأحداث) و مجموعةٍ من الإجراءات (سواءٌ أكانت إجراءاتٍ ثابتةً أم إجراءاتٍ حرةً أو مزيجاً من النوعين)، و يتم استدعاء هذه الإجراءات عند وقوع أيٍ من تلك الأحداث المرتبطة. و يتم الربط بإحدى الصيغ التالية:

صيغة 1 :

متعهد الاستنساخ التغطية الجدة اسم.الحدث(أسماء مُرسَلات الحدث بنفس ترتيب كتابتها فيه. و كتابتها هنا اختيارية): \ أسماء الإجراءات كل اسم في سطر منفرد /

الاستنساخ:

الصفة	الشرح
مشترك	لا يستطيع التعامل مع متغيرات الكائنات إلا بنسبتها إلى الكائن الحاوي لها. و يجب أن يكون نوع المتعهد مشترك مادام الحدث نفسه مشتركاً
بدون صفة (الافتراضية)	هو المتعهد المستنسخ العادي الذي لا يتعامل إلا مع الأحداث المستنسخة الخاصة بالكائن الحاوي له

التغطية:

الصفة	الشرح
سقف	لا يمكن تغطيته في الأصناف التي ترث صنفه
بدون صفة (الافتراضية)	هو المتعهد العادي الذي يقبل التغطية

الجدة:

الصفة	الشرح
جديد	هو متعهدٌ له نفس الاسم الخاص بمتعهدٍ آخرٍ في صنفٍ يرثه صنفه. و يقوم هو بالتعديل في بنائه. أي أنه يغطى متعهداً آخراً في صنفٍ أبٍ لصنفه (يمكن استخدام هذه الصفة حتى إذا تم استخدام الصفة مجرد من قبل؛ نظراً لأنه في كل الأحوال

	لن يرى البرنامج إلا المتعهد الجديد المُغطِّي للقديم)
بدون صفة (الافتراضية)	هو المتعهد العادي الذي ليس هناك متعهدٌ آخرٌ له ذات اسمه في الصنف أو الأصناف التي يرثها صنفه

و القيم المُرسَلة من الحدث يمكن للمبرمج التعامل معها بإحدى ثلاث طرق:

- الإهمال التام و عدم الاستخدام. و يمكنه في هذه الحالة عدم كتابتها في القوسين التاليين لاسم الحدث من الأصل،
- استخدامها جميعاً أو بعضها بأي شكلٍ من الأشكال داخل أقواس معاملات الإجراءات و النداءات في صلب المتعهد، و يجب عندها كتابة أنواعها بنفس الترتيب الواردة به من الحدث لا بترتيبٍ مختلف، أما الأسماء فله أن يغيرها كما يشاء.

مثال 1 : يمكننا كتابة المتعهد التالي للحدث **تغيرحساب** الموجود في مثال 1 عن الأحداث:

```
متعهد سقف تغير.حساب:
        عرض.القائمة()
```

أو يمكننا كتابة المتعهد التالي:

```
متعهد تغير.حساب(رقم م  رقم ن):
        عرض.القائمة()
        عرض.الأسعار(ن  م+10  "قائمة الأسعار")
```

حيث نفترض هنا أن (عرض.الأسعار) هو إجراءٌ يأخذ ثلاث مُعامِلات، أولهن و ثانيهن من النوع رقم، و الثالث من النوع نص.

صيغة 2:

```
متعهد الاستنساخ التغطية الجدة حدث 1 (أسماء المُرسَلات) و حدث 2(أسماء المُرسَلات) و .... :
        أسماء الإجراءات
```

مثال:

```
متعهد تغير.حساب و حذف.حساب و إنشاء.حساب :
        عرض.القائمة()
```

الخلاصة

بمقارنة القواعد الحالية في إبداع لدعم الأحداث و مُتعهّداتهن بالشكل القديم (و الذي أوردتُه فيما سبق بالتفصيل): نجد أن الشكل الحالي يتفوق علي الشكل الأقدم بمنتهي الوضوح؛ و من أسباب ذلك:

- الشكل القديم يعتمد علي وجود مكوناتٍ قائمةٍ بذاتها لتؤدي المهام التي نرغب في القيام بها. و هذه المكونات لا تصلح للقيام بمهامٍ أخري سوي تلك المهام التي وُضِعَت لأجلها. بينما في الشكل الحالي فإن الاعتماد يقع علي الإجراءات الحرة و الإمكانيات المختلفة في التعامل معهن، و تمتاز الإجراءات الحرة بأن لها استخداماتٍ متنوعةً جداً و ليست فقط ذات مهمةٍ واحدةٍ تقوم بها.
و هذا (مع تكراره في أمورٍ أخري في تصميم اللغة) يؤدي بالضرورة إلي تقليص حجمها نظراً لأن المُكوِّن الواحد يقوم بأداء أكثر من مهمة، بينما يحدث العكس في الحالة الأخري لأن لكل مهمةٍ منفردةٍ نقوم بضم المُكونات الجديدة و تعقيد القواعد، بل و رأيتُ من لغات البرمجة ما فيها أكثر من قاعدةٍ واحدةٍ للقيام بذات المهمة (و انظروا إن شئتم إلي visual basic.net بتمعن؛ لتروا ما أقصده بالحجم السرطاني للقواعد المتشابهة[1]) !

- الشكل الحالي يزيد من خاصية "التمازجية orthogonality"[2] التي نعني بها القدرة علي مزج المكونات المختلفة في لغة البرمجة مع بعضها البعض بشكلٍ سهل، بما يؤدي إلي زيادة إمكانيات اللغة المتاحة للمبرمج مع البقاء سهلة، و هو ما يعني قدراتٍ أعلي و تناسباً أكبر مع مهارات المبرمجين.

- الشكل الحالي يسهل الأمر بكثيرٍ علي المبرمجين العاملِين علي بناء مُفسِّر أُبْدِع؛ ففي الشكل القديم كان يجب مراعاة الأمور التالية في العمل:
  - بناء أكواد تعريفات الأحداث و المتعهدات، و التأكد من أن تلك التعريفات موجودةٌ في الأماكن المسموح فيها بذلك فقط، بحيث يمكن أن يكون التعريف موجوداً داخل الملف الرئيس مباشرةً، و/أو داخل صنفٍ من الأصناف ... إلخ، و لكن ليس في أي مكانٍ آخر،

1 ربما أقوم في المستقبل بمشيئة الله تعالي بكتابة مقالٍ أو سلسلة مقالاتٍ أتحدث فيها عن الأسباب التي تجعلني أوقن أن كثيراً من مُكوِّنات لغة الـ visual basic .net مُكرَّرة، و أنه كان يمكن تقليل حجم قواعد اللغة بشكلٍ كبيرٍ للغاية بحذف و/أو دمج كثيرٍ من القواعد المتشابهة الوظيفة و التكوين.

2 استخدام تعبير "التمازجية" كترجمةٍ لتعبير orthogonality هو من اجتهادي الخاص، و لا أدري هل استخدمه أحدهم من قبل لذات الغرض أم لا. و قد اخترتُ هذه الترجمة لأنها تُعبِّر عن الفكرة المقصودة بشكلٍ أوضح من حيث أنها "القدرة علي مزج التعبيرات المختلفة في لغة البرمجة بشكلٍ سهلٍ للحصول علي إمكانياتٍ أعلي".

- ○ بناء أكواد التحقق من أن وقوع حالات الاستخدام usage cases اللاتي يُصاحبن هذه المكونات تُوجَد في الأماكن المسموح بكتابتها فيها فقط.
- ○ كتابة أكواد التحقق من القواعد التي تخضع لها هذه المكونات في عملية الوراثة، مثل عدم تغطية أحدها حينما لا يكون قابلاً للتغطية، أو أن يكون للمكون الذي يغطيه نفس الخصائص التي له بحيث يكونان متماثلين من حيث الواجهة الخارجية.

و ربما غيرهن من القواعد التي تأكل الوقت و الجهد.
بينما في الحالة الأحدث فإن الأمر له نفس قواعد التعامل مع الإجراءات الحرة بدون زيادةٍ أو نقصان، أي أنه ما إن يتم دعم التعامل مع الإجراءات الحرة في أُبْدِع حتي ينتهي الأمر كله بدون الحاجة لفعلٍ شيءٍ جديد.
و هو ما يعني قدرةً أكبر علي التركيز علي تحسين بقية الأمور في هيكل عمل المُفسِّر، مثل تحسين تكوين/التعامل مع عُقَد التنفيذ executing nodes، و كذا طرق عمل تحسيناتٍ optimizations عليها بحيث يُصبح تنفيذ الكود أسرع و أقوي. و غيرهن من الأمور التي يمكن إنفاق الوقت الذي تم توفيره عليهن.

- الشكل الجديد من الطرق التي تؤدي لتصغير حجم قواعد اللغة بما يؤدي إلي تقليل منحني تعلُّم اللغة نفسها، صحيحٌ أن تعليم استخدام الإجراءات الحرة له منحني تعلمٍ في حد ذاته؛ لأن الإجراءات الحرة ليست بالأمر الهين أو الخفيف، و لكن يظل ذلك المنحني أصغر بكثيرٍ جداً من منحني تعلُّم كل تلك القواعد التي أسلفنا ذكرها في هذا المقال و المقال السابق له.